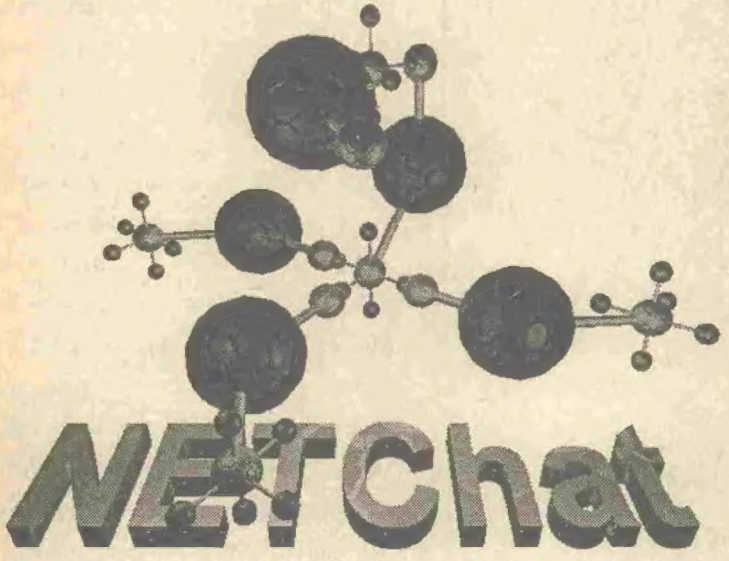
ประพนธ์ อัศวภาณุวัฒน์

SOFT MAIL

Communications : NetChat (ตอนที่ 1)

ลองเขียนโปรแกรมสื่อสาร ระหว่างเครื่องคอมพิวเตอร์ เพื่อเป็นแบบอย่างในการแลกเปลี่ยน ข้อมูลระหว่างเครื่องคอมพิวเตอร์



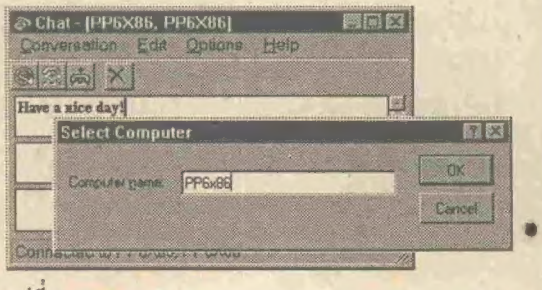
สวัสดีครับ ในปัจจุบันการนำคอมพิวเตอร์ไปประยุกต์ใช้งานในระบบธุรกิจต่าง ๆ มีจำนวนเพิ่มมากขึ้น อัตราส่วนการเพิ่มขึ้นนี้รวมทั้งการเพิ่มของการทำงานแบบกลุ่มงาน (Working Group) มากขึ้นด้วย สาเหตุเนื่องจากธุรกิจในปัจจุบันมีการแข่งขันกันมาก ผู้ที่มีความชำนาญและผู้ที่มีข้อมูลอยู่จำนวนมากย่อมได้เปรียบ แต่การมีข้อมูลจำนวนมากซึ่งเปรียบเสมือนข้อมูลดิบอย่างเดียวไม่ได้เป็นสิ่งช่วยส่งเสริมความก้าวหน้าขององค์กร การปรับตัวขององค์กรเพื่อนำข้อมูลที่มีอยู่มาใช้ประโยชน์ให้ได้มากที่สุดจึงเป็นสิ่งสำคัญยิ่ง และการแลกเปลี่ยนข้อมูลในหน่วยงานต่าง ๆ ขององค์กรจะช่วยเพิ่มศักยภาพของข้อมูลที่มีอยู่เป็นอย่างดี

เช่นเดียวกันกับการแข่งขันทางด้านธุรกิจ การพัฒนาโปรแกรมแบบปิดสำหรับผู้ใช้เดียว (Single User) เริ่มมีบทบาทลดน้อยลง ตรงกันข้ามกับการพัฒนาโปรแกรมแบบเปิดที่ยอมให้โปรแกรมอื่นหรือผู้ใช้อื่นสามารถเข้าแลกเปลี่ยนข้อมูลได้กลับมีบทบาทมากขึ้น โปรแกรมส่วนใหญ่เริ่มมีการปรับตัวเพื่อที่สามารถแลกเปลี่ยนข้อมูลกันได้ การแลกเปลี่ยนข้อมูลกันภายในระบบคอมพิวเตอร์เครื่องเดียวกันทำได้หลายวิธี เช่น OLE (Object Linking and Embedding), File Mapping, DDE (Dynamic Data Exchange) ฯลฯ สำหรับการแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์สามารถทำได้เช่นกันด้วย Network Protocol ต่าง ๆ เช่น NetBEUI, IPX/SPX และ TCP/IP เป็นต้น

ตามที่ได้เกริ่นไว้ในฉบับที่แล้ว ในฉบับนี้เราจะลองเขียนโปรแกรมสื่อสารระหว่างเครื่องคอมพิวเตอร์เพื่อเป็นแบบอย่างในการแลกเปลี่ยนข้อมูลระหว่างเครื่องคอมพิวเตอร์ โดยผมขอยกตัวอย่างโปรแกรมที่มีอยู่ก่อนแล้วในชุดระบบปฏิบัติการวินโดวส์ คือโปรแกรม WinChat (WinChat.exe) เป็นโปรแกรมต้นแบบ

WinChat.exe

WinChat (รูปที่ 1) เป็นโปรแกรมสำหรับสื่อสารระหว่างเครื่องคอมพิวเตอร์ ทำให้ผู้ใช้ที่เครื่องคอมพิวเตอร์ต้นทางและเครื่องคอมพิวเตอร์ปลายทางสามารถคุย (พิมพ์) โต้ตอบกันได้ โปรแกรม WinChat ใช้การแลกเปลี่ยนข้อมูลแบบ NDDE (Network Dynamic Data Exchange) ทำงานบน NetBEUI Protocol ส่งข้อมูลผ่านระบบ Network ถึงกัน



รูปที่ 1

WinChat เป็นชุดโปรแกรมมาตรฐานสำหรับระบบ Windows 3.X, Windows NT (คือติดตั้งมาพร้อมกันกับระบบ) สำหรับระบบ Windows 95 ได้แยกเป็นชุดโปรแกรมติดตั้งต่างหาก อยู่ในไดเรคทอรี \OTHER\CHAT ในแผ่นโปรแกรม CD-ROM ติดตั้ง Windows 95 วิธีการติดตั้งสามารถค้นหาได้จากไฟล์ WinChat.Txt ที่ให้มาพร้อมกับโปรแกรม

จากการทดลองใช้งาน เราสามารถที่จะสรุปประสิทธิภาพโดยรวมของโปรแกรม WinChat ดังนี้คือ

1. ส่งข้อมูลชนิด Text โต้ตอบกันได้

2. ใช้ NetBEUI เป็น Network Protocol มาตรฐาน ทำให้สามารถติดต่อกับ Windows ในเวอร์ชันต่างกันได้ (ต้องติดตั้ง NetBEUI Protocol)

3. ถ้าเครื่องคอมพิวเตอร์ปลายทางได้เคยทำการรัน NDDE Service แล้ว เมื่อมีการเรียกจากเครื่องต้นทางจะสามารถรับการเรียกเข้าโดยอัตโนมัติ (ไม่ต้องรันโปรแกรม WinChat ทิ้งไว้)

4. สามารถใช้งานแบบกลุ่ม (ผู้ใช้หลายคนในหน้าต่างเดียวกัน) โดยมีเงื่อนไขคือ

- ผู้ที่ทำการติดต่อ (Dial) ต้องเป็นคนเดียวกัน กล่าวคือ เมื่อเครื่อง A (Caller) ทำการติดต่อไปยังเครื่อง B (Receiver) แล้ว เครื่อง C จะสามารถเข้าร่วมกลุ่มกับเครื่อง AB ได้ก็ต่อเมื่อเครื่อง A ทำการติดต่อไปยังเครื่อง C เท่านั้น
- เครื่องที่ทำหน้าที่เป็น "Receiver" แล้ว จะไม่สามารถเรียกหรือทำการติดต่อไปยังเครื่องอื่น ๆ ได้อีก นอกจากสร้างกลุ่มใหม่

เตรียมการ

หลังจากสรุปความสามารถของโปรแกรม WinChat แล้ว ผมคิดว่าในการพัฒนาโปรแกรมตัวนี้ขึ้นใหม่ ด้วย Delphi เราสามารถปรับปรุงประสิทธิภาพของโปรแกรมได้เป็น

1. สามารถส่งข้อมูลได้ทั้งชนิด Text และ Binary การปรับโปรแกรมให้สามารถส่งข้อมูลได้ทั้งชนิด Text และ Binary ทำให้ได้ประโยชน์เพิ่มขึ้นคือ ไม่จำกัดชนิดข้อมูลที่ต้องการส่งให้กับเครื่องปลายทาง ไม่ว่าจะเป็นข้อมูลชนิด Text หรือแฟ้มข้อมูลต่าง ๆ และผลพลอยได้อีกอย่างคือ ทำให้ออกแบบชุดโครงสร้างคำสั่งได้ง่าย และมีระบบมากขึ้น

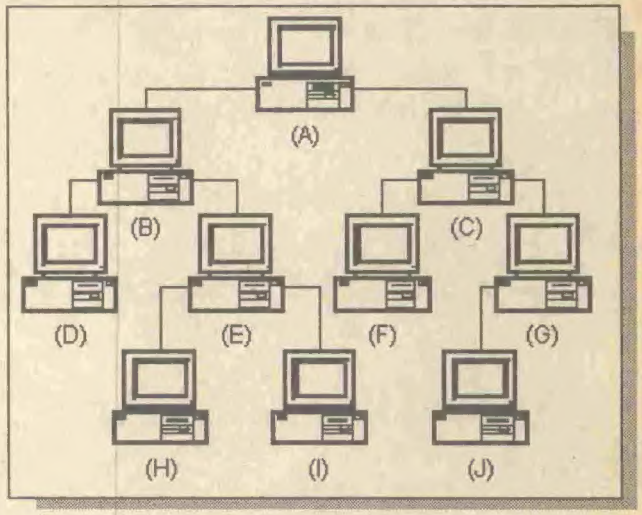
2. ใช้ TCP/IP Protocol เป็นโปรโตคอลมาตรฐาน สำหรับการติดต่อของโปรแกรม สาเหตุที่ต้องใช้ TCP/IP Protocol เพื่อสนับสนุนการส่งข้อมูลชนิด Binary (ที่ต้องกล่าวเช่นนี้เพราะผมเองเคยมีประสบการณ์ในการเขียนโปรแกรมส่งข้อมูลด้วย NDDE บน Windows 3.11 for Workgroup และประสบปัญหาคือไม่สามารถส่งข้อมูลที่มีรหัสข้อมูลต่ำกว่า 32 (ASCII 32)) และที่สำคัญสำหรับชุดโปรแกรม Delphi เองก็มี Components ที่สนับสนุนการส่งข้อมูลด้วย TCP/IP Protocol ทำให้ง่ายต่อการพัฒนาระบบ

3. สามารถติดต่อกันในลักษณะ Personal Mode หรือ Conferrence Mode เราจะเรียกคอมพิวเตอร์ที่ติดต่อเข้าสู่กลุ่มของเราว่า "Node" และการติดต่อกันภายในกลุ่มจะมีทางเลือก 2 แบบคือ Personal Mode หมายถึง Node แต่ละตัวสามารถเลือกติดต่อกันเฉพาะจุด หรือสามารถเลือกติดต่อเฉพาะกลุ่ม Node ที่เลือกได้ และ Conferrence Mode หมายถึงติดต่อกับทุก Node ที่ติดต่อเข้ามาในระบบขณะนั้น

4. สามารถทำหน้าที่เป็น Server เพื่อให้เครื่องอื่นติดต่อเข้ามา และในขณะเดียวกันต้องทำหน้าที่เป็น Clients เพื่อติดต่อไปยังเครื่องอื่น การทำให้โปรแกรมเป็นทั้ง Server และ Clients ในตัว ทำให้สามารถติดต่อกันในลักษณะ Tree เป็นผลให้ระบบไม่จำกัดการติดต่อ ตัวอย่างดังรูปที่ 2 เครื่อง A เป็นเครื่องแรกที่รันระบบ จึงทำหน้าที่เป็น Server อย่างเดียว สำหรับเครื่องอื่นที่อยู่ในระบบอันได้แก่เครื่อง B, C, E และ G จะทำหน้าที่เป็นทั้ง Server และ Clients เช่นเครื่อง B ทำหน้าที่เป็น Clients ของเครื่อง A และเป็น Server ของเครื่อง D และ E เป็นต้น สำหรับเครื่อง D, F, H, I, J ในรูปจะทำหน้าที่เป็น Clients อย่างเดียว จนกว่าจะมีการติดต่อจากเครื่องอื่น ๆ

5. ความต้องการของระบบ โปรแกรมชุดนี้ผู้เขียนได้พัฒนาขึ้นด้วย Delphi 3.0 บนระบบปฏิบัติการวินโดวส์ 95 ด้วยเครื่อง Pentium-100 มีหน่วยความจำขนาด 32 เมกะไบต์ และรันทดสอบด้วย RAM ขนาด 16 เมกะไบต์ พบว่าสามารถทำงานได้เป็นปกติ ผู้เขียนจึงขอกำหนดความต้องการของระบบดังนี้

- เครื่องคอมพิวเตอร์ระดับ Pentium-100 ขึ้นไป
- ระบบปฏิบัติการวินโดวส์ 95, วินโดวส์ NT 4.0 หรือสูงกว่า
- ระบบ Network อันได้แก่ Network Card และสายสัญญาณ
- โปรแกรม Delphi 3.0 Professional หรือ Delphi 3.0 Clients/Server สาเหตุที่ต้องเลือกชุดโปรแกรม 2 เวอร์ชันนี้เนื่องจากทั้งสองชุดมีชุด Internet Components ที่ต้องใช้ในการพัฒนาโปรแกรม คือ TCP ActiveX Control ซึ่งจะกล่าวถึงในส่วนต่อไป



รูปที่ 2

6. ข้อตกลงเพิ่มเติมสำหรับระบบนี้

● ก่อนดำเนินการในขั้นต่อไปผมขออนุญาตตั้งชื่อให้โปรแกรมก่อนเพื่อให้ง่ายต่อการอ้างถึง "NetChat" ด้วยคุณสมบัติของตัวโปรแกรมเองน่าจะเป็นชื่อที่เหมาะสม

● ระดับการติดต่อสำหรับเครื่องทุกเครื่องที่ติดต่อกับเครื่องแม่เดียวกัน ผมจะเรียกว่าอยู่ใน "กลุ่ม" (Group) เดียวกัน และเครื่องทุกเครื่องที่ติดต่อถึงกันด้วยโปรแกรมนี้ได้ผมจะเรียกว่าอยู่ใน "ระบบ" (System) เดียวกัน

● สมาชิกในระบบจะทำการลงทะเบียนเพื่อติดต่อเข้ากลุ่มได้เพียงกลุ่มเดียวต่อครั้ง หมายถึงเครื่องทุกเครื่องที่ต้องการใช้งานภายในระบบจะมีเครื่องแม่ได้เพียงเครื่องเดียว เช่น เครื่อง A ไม่สามารถติดต่อลงทะเบียนเป็นสมาชิกในกลุ่มของเครื่อง C ในขณะที่เป็นสมาชิกของเครื่อง B

● จำนวน Connection สูงสุดสำหรับเครื่องทุกตัวเท่ากับ 7 แบ่งออกเป็น 2 Connection สำหรับใช้ในระบบ และอีก 5 Connection สำหรับรับการติดต่อจากระบบอื่น ๆ (รับการติดต่อจากเครื่องลูกได้ 5 เครื่อง)

● กรรมวิธีการส่งชุดข้อมูล (Package Message) ในระบบ จะส่งชุดข้อมูลแบบกระจาย (Boardcast) เข้าสู่ระบบที่ติดต่ออยู่ทั้งหมด และระบบที่ถูกระบุให้เป็นผู้รับชุดข้อมูลเท่านั้นที่มีสิทธิในการตีความ (Dispatch) ชุดข้อมูลนั้น ๆ

● การติดตั้งเพื่อทดสอบใด ๆ ควรจะกระทำบนระบบ Network ที่แยกออกจากระบบที่ใช้งานจริง เพื่อป้องกันปัญหา Side effected ต่าง ๆ

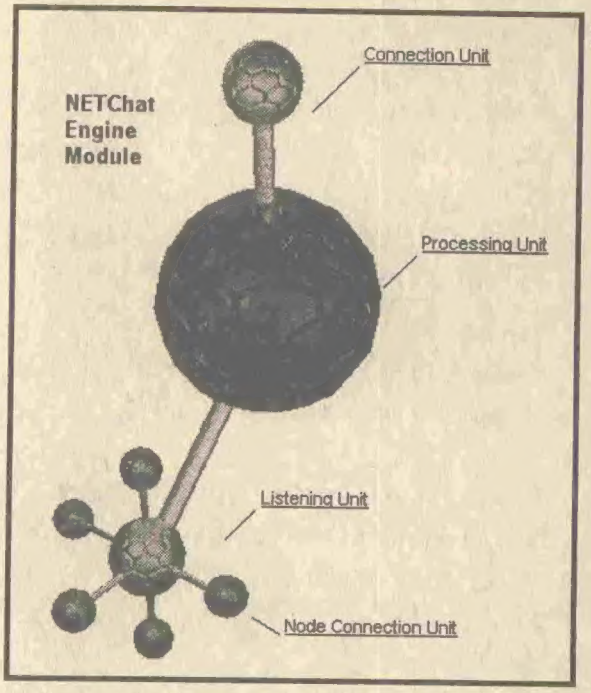
ออกแบบเอ็นจิน (Engine)

เมื่อทราบจุดมุ่งหมายของระบบแล้ว ก็มาถึงขั้นตอนการออกแบบระบบการทำงานของโปรแกรม ผมขอเรียกส่วนระบบการทำงานของโปรแกรมว่า "เอ็นจิน" (Engine) การออกแบบเอ็นจินจำเป็นต้องคำนึงถึงข้อจำกัดต่าง ๆ ในระบบปฏิบัติการที่ใช้ เช่น ระบบวินโดวส์ 95 รับรองการติดต่อได้ 10 Connections ต่อเครื่อง ชนิดของ CPU ที่ใช้ หรือขนาดของ RAM มาตรฐานที่เครื่องอื่น ๆ ควรมี เป็นต้น เพราะสิ่งเหล่านี้เป็นปัจจัยพื้นฐานสำหรับเสถียรภาพของระบบ

ระบบเอ็นจินสำหรับโปรแกรมนี้แบ่งออกเป็น 4 ส่วนหลัก ดังรูปที่ 3 คือ

● Connection Unit เป็นส่วนที่ใช้สำหรับติดต่อไปยัง Listening Unit ของระบบอื่น ๆ ถ้าติดต่อสำเร็จจะส่งเมสเสจไปลงทะเบียนเป็นยูสเซอร์ในกลุ่มนั้น ๆ

● Listening Unit เป็นส่วนรองรับการติดต่อจากระบบอื่น ๆ คือ เมื่อได้รับการติดจาก Node ใด ๆ แล้ว Listening Unit จะทำการสร้าง Node Connection Unit ขึ้นเพื่อให้รับการติดต่อจาก Node ที่ติดต่อเข้ามา และทำการรอการติดต่อจาก Node อื่น ๆ ต่อไป



รูปที่ 3

● Node Connection Unit เป็นหน่วยรับผิดชอบการติดต่อกับระบบอื่น ๆ ที่ติดต่อเข้ามา โดยจะดูแลการรับส่งเมสเสจต่าง ๆ

● Processing Unit เป็นส่วนหลักของโปรแกรมทำหน้าที่เชื่อมโยงส่วนต่าง ๆ เข้าด้วยกัน รวมทั้งควบคุมระบบการทำงานของเอ็นจินในส่วนต่าง ๆ เช่น ควบคุมยูสเซอร์ที่ติดต่อเข้าระบบ และแสดงผลออกหน้าจอ เป็นต้น

ขั้นตอนการทำงานของโปรแกรม

เมื่อรันโปรแกรม ระบบต่าง ๆ ของโปรแกรมจะเริ่มทำงานตามลำดับดังรูปที่ 4 มีรายละเอียดดังต่อไปนี้

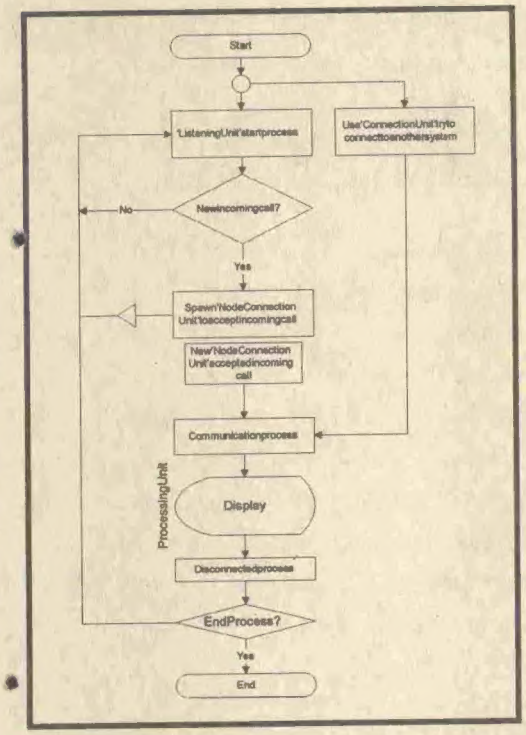
● ส่วน "Listening Unit" จะถูกเรียกใช้เป็นส่วนแรกเพื่อรอการติดต่อจากเครื่องอื่น ๆ

● ถ้ามีระบบอื่นติดต่อเข้ามา "Node Connection Unit" จะถูกเรียกขึ้นทำงานเพื่อรับการติดต่อ และ "Listening Unit" จะรอการติดต่อจากระบบอื่น ต่อไป

● ในขณะเดียวกันถ้าต้องการติดต่อไปยังเครื่องอื่น ระบบจะใช้ "Connection Unit" เพื่อติดต่อไปยังเครื่องอื่น

● ถ้าติดต่อสำเร็จ ไม่ว่าจะเป็น "Node Connection Unit" หรือ "Connection Unit" จะดำเนินการในส่วน "Processing Unit" คือ แปลเมสเสจ (Dispatch Message) และดำเนินการตามคำสั่งที่แปลได้ เช่น แสดงผลต่อไป

● ถ้ายังไม่ยุติโปรแกรม "Listening Unit" จะรอการติดต่อจากเครื่องอื่น และ "Processing Unit" จะทำการประมวลผลข้อมูลชุดใหม่



รูปที่ 4

กระบวนการเชื่อมการติดต่อ

กระบวนการเชื่อมการติดต่อจะสมบูรณ์ได้ก็ต่อเมื่อสมาชิกใหม่ผู้ร้องขอติดต่อ ทำการติดต่อไปยังเครื่องอื่น ๆ ที่อยู่ภายในระบบ และหลังจากได้รับการตอบรับจากเครื่องที่ทำการติดต่อไป จะต้องส่งชุดข้อมูล Register Package ไปลงทะเบียนเป็นสมาชิกใหม่

หลังจากลงทะเบียนเป็นสมาชิกใหม่แล้ว จะได้รับรายชื่อสมาชิกในระบบทั้งหมด พร้อมกับได้รับการแนะนำไปยังสมาชิกอื่น ๆ ให้รู้จัก

สร้างโปรแกรม

ในขั้นตอนการออกแบบเอ็นจินของโปรแกรม ผมพยายามสร้างโมเดล (Model) โดยแยกเป็นส่วนต่าง ๆ เพื่ออธิบายให้ผู้อ่านมองเห็นภาพและเข้าใจลำดับการทำงานของระบบ ซึ่งเมื่อแปลงเป็นโปรแกรมแล้วจะเหลือให้เป็นรูปธรรมเพียง 2 ส่วนคือ ฟอร์มหลัก (uNetChat.pas) เป็นส่วนที่บรรจุระบบ 3 ระบบคือ "Connection Unit", "Listening Unit" และ "Processing Unit" เข้าไว้ด้วยกัน และฟอร์มย่อย (uCntSckt.pas) เป็นส่วนของระบบ Node Connection Unit

uNCmisc.pas

ผมได้กำหนดรูปแบบข้อมูลขึ้นใหม่บางส่วน รวมทั้งสร้างกิจกรรมที่ต้องใช้ในทั้งสองฟอร์ม แยกไว้ในยูนิตไฟล์ (uNCmisc.pas) เพื่อให้ง่ายต่อการเรียกใช้ และผู้อ่านที่สนใจสามารถนำไปแก้ไขเพิ่มเติม โดยเพิ่มกิจกรรมและเมสเสจต่าง ๆ ไว้ในไฟล์นี้ได้ทันที ส่วนของโปรแกรมที่สำคัญมีดังต่อไปนี้

ค่าคงที่ และ แบบข้อมูล

```
// ระบุจำนวน Client ที่สามารถ Connect เข้าสู่ระบบได้
MAX_CNT_SCKT       = 5;
// ยูสเซอร์เมสเสจที่ใช้สื่อสารภายในโปรแกรม
wm_IamClose = wm_User + $1000;
        // แจ้งการปิดฟอร์ม
wm_UserConnect = wm_User + $1001;
        // แจ้งมี New user ติดต่อเข้าสู่ระบบ
wm_IntroductMember2User = wm_User + $1003;
        // แนะนำยูสเซอร์ที่มีอยู่ในระบบให้กับ New User
wm_ContentFromClient     = wm_User + $1004;
        // ส่งข้อมูลจากฟอร์มย่อยให้ฟอร์มหลัก
// ชนิดการลงทะเบียน
REG_LOGIN    = $1001; // ติดต่อเข้าสู่ระบบ
REG_LOGOUT   = $1002; // ยกเลิกการติดต่อ
REG_INTRODUCT      = $1003; // แนะนำยูสเซอร์คนใหม่ให้กับกลุ่ม
// ชนิดเมสเสจที่ใช้รับส่งระหว่างโปรแกรม (สามารถเพิ่มเติมเมสเสจชนิดใหม่ได้)
MSG_SYS_REG = $1011; // เมสเสจระบุการลงทะเบียนเข้าสู่ระบบ
MSG_MEMBER_LIST    = $1012; // เมสเสจระบุรายชื่อสมาชิกในกลุ่มเดียวกัน
MSG_CONTENT = $1013; // เมสเสจชนิดข้อความ
```

ชุดข้อมูล (Package) ที่ใช้ส่งระหว่างโปรแกรมมีรูปแบบคือ 2 ไบต์แรกจะเป็นชนิดเมสเสจ (MsgType) และ 2 ไบต์ต่อมาระบุขนาดของเมสเสจ (msgLen) มีรูปแบบ คือ

```
TmessageR = record
  msgType    : word;
  msgLen     : word;
  ..................   // ชุดข้อมูลใช้ขึ้นอยู่กับชนิดของเมสเสจ
end;
```

ชุดข้อมูล (Package Message) สำคัญที่ใช้ในระบบ ประกอบไปด้วย

```
Tregister2NetR = record
// ชุดข้อมูล การขอลงทะเบียนเข้าเป็นยูสเซอร์คนใหม่ในกลุ่ม
  msgType    : word;
// ชนิดเมสเสจ ระบุเป็น MSG_SYS_REG
  msgLen     : word;
// ความยาวของชุดข้อมูลนี้ สามารถระบุได้เท่ากับ sizeof( Tregister2NetR )
  regType    : word; // ชนิดการขอลงทะเบียน สามารถระบุได้เป็น
REG_LOGIN, REG_LOGOUT, REG_INTRODUCT
  regName    : Tstr50; // ชื่อของระบบที่ใช้ลงทะเบียน
  regIp      : Tstr50; // หมายเลข IP Address ที่ใช้ลงทะเบียน
  regId      : Double; // หมายเลข ID ที่ใช้ลงทะเบียน
end;
TnetMemberR = record // ชุดข้อมูลระบุรายชื่อสมาชิกในระบบ
    msgType : word; // ชนิดเมสเสจ ระบุเป็น MSG_MEMBER_LIST
    msgLen  : word; // ความยาวของชุดข้อมูล เท่ากับ sizeof
( TnetMemberR )
    memberName     : Tstr50; // ชื่อสมาชิก
  end;
TcontentBlockR = record // ชุดข้อมูล ส่ง Content
ให้กับสมาชิกที่ระบุ หรือภายในกลุ่มที่ติดต่อ
    msgType : word; // ชนิดเมสเสจ ระบุเป็น MSG_CONTENT
    msgLen  : word; // ความยาวของชุดข้อมูล เท่ากับ sizeof
( TcontentBlockR )
    senderName     : Tstr50; // ชื่อเครื่องที่ส่ง Content
    confMode       : boolean;
// flag ถ้าระบุเป็น TRUE จะใช้เป็น conference mode
    clientList     : T1KBS;
// ถ้าระบุ confMode เป็น FALSE จะต้องระบุรายชื่อเครื่องที่ต้องการส่ง Content ให้ไว้ที่นี่
    content : String; // ข้อความที่ต้องการส่ง
  end;
```

โปรแกรมย่อย

จากบทความก่อนหน้านี้ที่ผมนำเสนอ ผมได้นำเอาชุดกิจกรรมยอดนิยมของผมมาใช้ในโปรแกรม NetChat นี้ด้วยกัน 2 กิจกรรมคือ กิจกรรม Schedule และกิจกรรม Delay(msec : longint); ซึ่งจะไม่ขอกล่าวถึงอีก คงเหลือกิจกรรม GetPackageSize เป็นกิจกรรมใหม่ มีรายละเอียดดังนี้ครับ

procedure GetPackageSize(var data : T64KBS; curByte : word; var msgType, pckSize : word); เป็นกิจกรรมใช้สำหรับอ่านค่าชนิด msgType และขนาดของเมสเสจ pckSize ที่รับมาใน data แล้วส่งกลับให้กิจกรรมหลักเพื่อตีความ

Node Connection Unit (uCntSckt.pas)

ยูนิต uCntSckt.pas แสดงอยู่ในลิสติงที่ 1 เป็นฟอร์มสำหรับรับการติดต่อจากเครื่องอื่น โดยใช้ TCP ActiveX Control ใน Internet Palette เป็นออบเจ็กต์สำหรับรับการติดต่อ มีหน้าที่หลักคือ

รูปที่ 5

- เชื่อมการติดต่อระหว่างสมาชิกคนใหม่เข้ากับกลุ่ม
- รับการติดต่อลงทะเบียนจากสมาชิกคนใหม่เพื่อส่งให้สมาชิกในกลุ่มและระบบทราบ
- คอยส่งข้อมูลต่าง ๆ ให้กับสมาชิกทั้งหมด

สำหรับฟอร์มนี้ มีตัวแปรและกิจกรรมที่สำคัญคือ

ตัวแปร

txBufferV : OleVariant; ตัวแปรชนิด OLE Variant ใช้เป็น Buffer สำหรับส่งข้อมูล (Transmit) ให้กับ ActiveX Control

rxBufferV : OleVariant; ตัวแปรชนิด OLE Variant อีกตัวที่ใช้เป็น Buffer ในส่วนการรับข้อมูลจาก ActiveX Control

ลิสติงที่ 1

```
00001:  List 1.1) UCNTSCKT.PAS
00002:
00003:  type
00004:    TFcntSckt = class(TForm)
00005:      TCP1: TTCP;
00006:      Panel1: TPanel;
00007:      procedure FormCreate(Sender: TObject);
00008:      procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
00009:      procedure TCP1Close(Sender: TObject);
00010:      procedure TCP1DataArrival(Sender: TObject; bytesTotal: Integer);
00011:    private
00012:      { Private declarations }
00013:        txBufferV : OleVariant;
00014:        rxBufferV : OleVariant;
00015:        rxBuffer  : T64KBS;
00016:        txBuffer  : T2KBS;
00017:        clientReg : Tregister2NetR;
00018:      procedure InitVar;
00019:      procedure DispatchMessage( bytesTotal : integer );
00020:      procedure Unpack_System_Register( curByte, pckSize : word );
00021:      procedure Unpack_ClientContent( curByte, pckSize : word );
00022:    public
00023:      { Public declarations }
00024:        myInfo : TcntScktInfoR;
00025:      procedure SendPackage( pck : T2KBS; pckSize : word );
00026:    protected
00027:      procedure wmIntroductMember2User( var mmsg : TMessage ); message wm_IntroductMember2User;
00028:    end;
00029:    TFcntScktS = array[ 0..MAX_CNT_SCKT-1 ] of TFcntSckt;
00030:  var
00031:    FcntSckt: TFcntScktS;
00032:  implementation
00033:  {$R *.DFM}
00034:  //---------------------------------------------------
00035:  procedure TFcntSckt.InitVar;
00036:  var p : pointer;
00037:  begin
00038:    fillchar( myInfo, sizeof( TcntScktInfoR ), 0 );
00039:    fillchar( rxBuffer, sizeof( T64KBS ), 0 );
00040:    fillchar( txBuffer, sizeof( T2KBS ), 0 );
00041:    txBufferV:=VarArrayCreate( [ 0, sizeof( T2KBS )-1 ], varByte );
```

ลิสติงที่ 1 (ต่อ)

```
00042:    rxBufferV:=VarArrayCreate( [ 0, sizeof( T64KBS )-1 ], varByte );
00043:    p:=VarArrayLock( txBufferV );
00044:    fillchar( p^, sizeof( T2KBS )-1, 0 );
00045:    VarArrayUnlock( txBufferV );
00046:    p:=VarArrayLock( rxBufferV );
00047:    fillchar( p^, sizeof( T64KBS )-1, 0 );
00048:    VarArrayUnlock( rxBufferV );
00049:    fillchar( clientReg, sizeof( Tregister2NetR ), 0 );
00050:  end;
00051:  //-----------------------------------------
00052:  procedure TFcntSckt.wmIntroductMember2User( var mmsg : TMessage );
00053:  var listView : TListView; c : integer; pck : TnetMemberR; p : pointer;
00054:  begin
00055:    if ( mmsg.wparam<>handle ) then exit;
00056:    listView:=pointer( mmsg.lparam );
00057:    if ( listView=nil ) then exit;
00058:    for c:= 0 to listView.items.count-1 do
00059:    begin
00060:      fillchar( pck, sizeof( TnetMemberR ), 0 );
00061:      with pck do
00062:      begin
00063:        msgType:=MSG_MEMBER_LIST;
00064:        msgLen:=sizeof( TnetMemberR );
00065:        memberName:=listView.items.item[ c ].caption;
00066:      end;
00067:      VarArrayRedim( txBufferV, sizeof( TnetMemberR ) );
00068:      p:=VarArrayLock( txBufferV );
00069:      move( pck, p^, sizeof( TnetMemberR ) );
00070:      VarArrayUnlock( txBufferV );
00071:      TCP1.SendData( txBufferV );
00072:      Delay( 100 );
00073:    end;
00074:  end;
00075:  //-----------------------------------------
00076:  procedure TFcntSckt.SendPackage( pck : T2KBS; pckSize : word );
00077:  var p : pointer;
00078:  begin
00079:    VarArrayRedim( txBufferV, pckSize );
00080:    p:=VarArrayLock( txBufferV );
00081:    move( pck, p^, pckSize );
00082:    VarArrayUnlock( txBufferV );
00083:    TCP1.SendData( txBufferV );
00084:    Schedule;
00085:  end;
00086:  //-----------------------------------------
00087:  procedure TFcntSckt.DispatchMessage( bytesTotal : integer );
00088:  var curByte,pckSize, msgType : word;
00089:  begin
00090:    curByte:=0;
00091:    repeat
00092:      GetPackageSize( rxBuffer, curByte, msgType, pckSize );
00093:      case msgType of
00094:        MSG_SYS_REG : Unpack_System_Register( curByte, pckSize );
00095:        MSG_CONTENT : Unpack_ClientContent( curByte, pckSize );
00096:        else
00097:        begin
00098:          ShowMessage( 'Dispatch error [FcntSckt], msgType = 0x'+IntToHex( msgType,4 ) );
00099:        end;
00100:      end;
00101:      inc( curByte, pckSize+1 );
00102:    until ( curByte >= bytesTotal ) or ( pckSize=0 );
00103:  end;
00104:  //- Unpack Secsion ------------------------
00105:  procedure TFcntSckt.Unpack_System_Register( curByte, pckSize : word );
00106:  begin
00107:    move( rxBuffer[ curByte ], clientReg, pckSize );
00108:    caption:=clientReg.regName;
00109:    SendMessage(myInfo.netChatHdl,wm_UserConnect,handle,longint(@clientReg));
00110:  end;
00111:  procedure TFcntSckt.Unpack_ClientContent( curByte, pckSize : word );
00112:  begin
```

→

ลิสติงที่ 1 (ต่อ)

```
00113:    move( rxBuffer[ curByte ], txBuffer, pckSize );
00114:    SendMessage( myInfo.netChatHdl, wm_ContentFromClient, handle, longint( @txBuffer ) );
00115:    Schedule;
00116:  end;
00117:  //-------------------------------------
00118:  procedure TFcntSckt.FormCreate(Sender: TObject);
00119:  begin
00120:    InitVar;
00121:  end;
00122:  //- Connection Method ------------------------------
00123:  procedure TFcntSckt.TCP1Close(Sender: TObject);
00124:  begin
00125:    Panel1.caption:='Close connection';
00126:    WindowState:=wsNormal; Schedule;
00127:    SendMessage( myInfo.netChatHdl, wm_IamClose,
00128:      myInfo.cntScktNo, longint( @clientReg ) );
00129:    Schedule;
00130:  end;
00131:  procedure TFcntSckt.TCP1DataArrival(Sender: TObject; bytesTotal: Integer);
00132:  var p : pointer;
00133:  begin
00134:    TCP1.GetData( rxBufferV, VT_ARRAY or VT_UI1, bytesTotal );
00135:    p:=VarArrayLock( rxBufferV );
00136:    move( p^, rxBuffer, bytesTotal-1 );
00137:    VarArrayUnlock( rxBufferV );
00138:    DispatchMessage( bytesTotal-1 );
00139:  end;
00140:  end.
```

rxBuffer : T64KBS; เป็น Buffer ใช้บันทึกข้อมูลที่รับมาจาก rxBufferV เพื่อนำข้อมูลไปตีความ

txBuffer : T2KBS; Buffer สำหรับเก็บข้อมูลที่ต้องการส่งให้กับเครื่องอื่น ๆ ระบบจะนำข้อมูลจาก txBuffer ไปแปลงเป็นข้อมูลชนิด OLE Variant (txBufferV) แล้วจึงส่งให้กับ TCP ActiveX Control

clientReg : Tregister2NetR; เก็บข้อมูลของเครื่องที่ทำการติดต่ออยู่ในปัจจุบันด้วยชุดข้อมูล Tregister2NetR

TFcntScktS = array[0..MAX_CNT_SCKT-1] of TFcntSckt; กำหนด TFcntScktS ให้เป็น Class ชนิด Array เพื่อรองรับการเรียกเข้า 1 Node ต่อ 1 ออบเจ็กต์ (การใช้ Class ชนิด Array จำเป็นต้องยกเลิกการสร้างฟอร์มแบบอัตโนมัติ สามารถทำได้โดยเลือกเมนู Project → Obtions แล้วเลือกหน้า Form → เลือก TFcntSckt Class จาก Auto-create forms Listbox (ด้านซ้าย) ไปใส่ไว้ใน Available formsListbox (ด้านขวา) → กดปุ่ม OK)

ภายในโปรแกรมเราสามารถเรียกใช้ Form ที่ไม่ได้สร้างด้วยระบบ Auto-create forms โดย

```
if ( Form1=nil ) then Application.CreateForm
( Tform, Form1 );  Form1.Show;
```

และยกเลิก (ถาวร) ด้วยคำสั่ง

```
Form1.Release;  Form1:=nil;
```

กิจกรรมที่สำคัญ

procedure TCP1DataArrival(Sender: TObject; bytesTotal: Integer); เป็นกิจกรรมแจ้งว่ามีข้อมูลมาถึง มีการดำเนินการอ่านข้อมูลจาก Port ดังนี้

```
TCP1.GetData( rxBufferV, VT_ARRAY
or VT_UI1,bytesTotal );
p:=VarArrayLock( rxBufferV );
move( p^, rxBuffer, bytesTotal-1 );
VarArrayUnlock( rxBufferV );
DispatchMessage( bytesTotal-1 );
```

- ใช้กิจกรรม GetData อ่านข้อมูลชนิด VT_ARRAY or VT_UI1 ไปไว้ในตัวแปรชนิด OLE Variant (rxBufferV) ด้วยข้อมูลขนาด bytesTotal
- สร้างตัวแปร pointer (P) ให้ล็อกที่ OLE Variant ด้วยคำสั่ง VarArrayLock
- ทำการเคลื่อนย้ายข้อมูลจาก OLE Variant ไปยัง Buffer ที่ใช้งานจริง (rxBuffer)
- ยกเลิกการใช้ด้วยคำสั่ง VarArrayUnlock
- แปลชุดข้อมูลด้วยคำสั่ง DispatchMessage

procedure DispatchMessage(bytesTotal : integer); กิจกรรมสำหรับแปลชุดข้อมูลที่รับจาก TCP ActiveX Control โปรแกรมจะทำการวนรอบเช็ค Message Type (MsgType) และขนาดของ Package ว่าได้ถูกกำหนดไว้ในระบบหรือไม่ ถ้าถูกกำหนดไว้ จะส่งให้กิจกรรมที่ทำหน้าที่ Unpack ข้อมูลตรวจสอบข้อมูลต่อไป

```
      repeat
        GetPackageSize( rxBuffer, curByte,
msgType, pckSize );
        case msgType of
          MSG_SYS_REG : Unpack_System_Register
( curByte, pckSize );
          MSG_CONTENT : Unpack_ClientContent
( curByte, pckSize );
          else ShowMessage( 'Dispatch error ');
        end;
        inc( curByte, pckSize+1 );
      until ( curByte >= bytesTotal ) or
( pckSize=0 );
```

procedure Unpack_System_Register(curByte, pckSize : word); กิจกรรมสำหรับ Unpack ข้อมูล โปรแกรมจะทำการคัดลอกข้อมูลจาก rxBuffer ไปยังตัวแปร clientReg ด้วยขนาด pckSize แล้วส่งให้กับฟอร์มหลักเพื่อลงทะเบียน ด้วยการส่งให้กับฟอร์มหลักด้วยวินโดว์ยูสเซอร์เมสเสจ wm_UserConnect พร้อมกับผ่านค่าแอดเดรสของ clientReg ไปให้

```
move( rxBuffer[ curByte ], clientReg, pckSize);
SendMessage( myInfo.netChatHdl, wm_UserConnect,
handle, longint( @clientReg ) );
```

procedure Unpack_ClientContent(curByte, pckSize : word); กิจกรรมสำหรับ Unpack ข้อมูลชนิดข้อความ (Content) โดยส่งวินโดว์เมสเสจ wm_ContentFromClient ให้กับฟอร์มหลัก

procedure SendPackage(pck : T2KBS; pckSize : word); กิจกรรมสำหรับ pack ข้อมูลเป็น OLE Variant เพื่อส่งให้กับเครื่องอื่นผ่านทาง TCP ActiveX Control มีขั้นตอนดังนี้

```
VarArrayRedim( txBufferV, pckSize );
p:=VarArrayLock( txBufferV );
move( pck, p^, pckSize );
VarArrayUnlock( txBufferV );
TCP1.SendData( txBufferV );
```

- จองหน่วยความจำชนิด OLE Variant ด้วยขนาด pckSize ให้กับ txBufferV
- สร้างตัวแปร pointer (P) ให้ล็อกที่ txBufferV ด้วยคำสั่ง VarArrayLock
- ทำการเคลื่อนย้ายข้อมูลจาก Buffer ไปยัง OLE Variant (txBufferV)
- ยกเลิกการใช้ด้วยคำสั่ง VarArrayUnlock
- ส่งข้อมูล txBufferV ด้วยกิจกรรม SendData

procedure wmIntroductMember2User(var mmsg : TMessage); message wm_IntroductMember2User; กิจกรรมทำหน้าที่ส่ง (แนะนำ) รายชื่อของ Node ทั้งหมดที่ติดต่ออยู่ภายใต้เครื่องแม่เดียวกัน ให้กับยูสเซอร์ใหม่ที่ติดต่อเข้ามา

เนื้อหากำลังเข้มข้นนะครับ แต่เนื่องจากตัวโปรแกรมทั้งหมดค่อนข้างยาว ฉบับนี้คงต้องจบเพียงเท่านี้ครับ สำหรับในส่วนโปรแกรมหลักที่เป็นส่วนสำคัญที่สุด และขั้นตอนการนำไปใช้งาน คงต้องติดตามภาค 2 ในฉบับหน้า

ก่อนจากกัน ผมได้รับการสอบถามจากท่านผู้อ่านที่สนใจเกี่ยวกับการโปรแกรมให้เครื่องคอมพิวเตอร์สามารถรับส่งข้อมูลด้วย Serial Port (COM1, COM2) ด้วยภาษาปาสคาลนั้น เนื่องจากท่านผู้อ่านยังคงพัฒนาโปรแกรมบนระบบดอสและตัวผมเองก็ทิ้งการโปรแกรมบนดอสมาค่อนข้างนาน คงต้องกราบขอความกรุณาจากท่านผู้รู้ที่ทราบการเขียนโปรแกรมในส่วนนี้ช่วยเขียนเป็นวิทยาทานให้ด้วยครับ สำหรับการใช้งานบนระบบวินโดวส์นั้นผมพอมีประสบการณ์บ้าง ด้วยการโปรแกรมบนวินโดวส์ 3.x, วินโดวส์ 95 คงต้องขอเวลาเรียบเรียงสักระยะเนื่องจากวินโดวส์ 16 บิตและ 32 บิตมีการใช้งานต่างกันโดยสิ้นเชิง

สวัสดีครับ

หมายเหตุ : ผู้ที่สนใจซอร์สโค้ดของบทความนี้ สามารถใช้บริการซอฟต์เมล์จากตอนที่ 2 ในฉบับถัดไป

สำหรับท่านผู้อ่านที่สนใจในบทความของผม แล้วมีปัญหาไม่สามารถรันโปรแกรมซอร์สโค้ดได้นั้น ผมได้ติดต่อเรียบร้อยแล้ว ปรากฏว่าท่านมีปัญหาเกี่ยวกับการคอมไพล์โปรแกรม ซึ่งผมได้กำหนด output ไดเรคทอรีเป็นไดรฟ์ F:\OUTPUT แต่เครื่องของท่านมีเพียงไดรฟ์ C:\ จึงเกิดปัญหาไม่สามารถคอมไพล์โปรแกรมได้ ผมได้ชี้แจงและแก้ไขปัญหานี้สำหรับบทความตั้งแต่ชุดนี้เป็นต้นไปเรียบร้อยแล้ว

วิธีการแก้ไขปัญหาสำหรับบทความชุดก่อนหน้านี้ ทำได้โดย

1. รันโปรแกรม Delphi
2. เลือกเมนู Project → Options
3. เลือกหน้า Directories / Conditions
4. ในช่อง Output Directory : ให้ลบ "F:\OUTPUT" ออก หรือเปลี่ยนเป็น Directory ที่มีอยู่จริง
5. กดปุ่ม [OK]